Project 2

컴퓨터소프트웨어학부
2017030519 홍유진

1. 실행 방법 및 환경
 - 실행 환경은 Linux 우분투로 실행
 - 실행 방법
  : 2_Parser폴더에 parser을 들어간 후 make 후 아래 명령어로 실행

```
root@hongyoujin:/home/hongyoujin/project1/2019_ELE4029_2017030519/2_Parser/parser
# ./cminus test.cm
```

2. 과제 내용
1) 실행 내용
 저번 프로젝트에서 Flex를 통해 만들었던 cminus.l scanner를 이용해 cminus.y를 만들어 parser의 역할을 수행하도록 한다. yacc(bison)을 통해 코드를 컴파일하고 이때 나오는 파일이 parser가 제 기능을 할 수 있도록 한다. 이때 y.tab.h와 y.tab.c가 생성된다.

(1) Makefile
 : 과제 pdf에 나와있는 Makefile을 통해 Makefile을 수정하였다. 이때, yacc cminus.y와 yacc -o y.tab.o cminus.y 명령어를 통해 Makefile에서 필요한 파일을 생성했다.

(2) main.c
 : 과제 pdf를 참조함. NO_PARSE FALSE , NO_ANALIZE TRUE, EchoSource, TraceScan FALSE로 수정

(3) globals.h

```c
typedef enum {StmtK, ExpK, DecK} NodeKind;
typedef enum {IfK,AssignK,CompK, SelK, IterK, RetK,RepeatK,ReadK,WriteK} StmtKind;
typedef enum {VarK, ArrVarK, OpK, CallK, ConstK, IdK} ExpKind;
typedef enum {TypeK, DecVarK, DecFunK, DecArrK, ParamVarK, ParamArrK} DecKind;

/* ExpType is used for type checking */
typedef enum {Void, Integer, Boolean} ExpType;

#define MAXCHILDREN 3

typedef struct treeNode
{
    struct treeNode* child[MAXCHILDREN];
    struct treeNode* sibling;
    int lineno;
    NodeKind nodekind;
    union { StmtKind stmt;
        ExpKind exp;
        DecKind dec; } kind;
    union { TokenType op;
        int val;
        char* name;
    //  int size;
        } attr;
    //int arrsize;
    ExpType type; /* for type checking of exps */
} TreeNode;
```

: yacc 디렉토리의 globals.h를 복사해서 가져온 후 수정함. 각각 필요한 enum을 kind에 추가 했다.

기존에 있던것에 NodeKind에 DecK 추가, StmtKind에 AssignK, ExpreK, CompK, SelK, IterK, RetK 추가, ExpK에 VarK, ArrVarK, CallK, ConstK를 추가 DecKind에 TypeK, DecVarK, DecFunK, DecArrK, ParamVarK, ParamArrK를 추가했다.
treeNode엔 새로운 kind인 DecKind dec;를 추가했다.

(3) util.h
DecKind를 위한 TreeNode * newDecNode(DecNode);를 추가했다.

(4) cminus.y
: yacc 디렉토리의 tiny.y를 복사해서 가져와 수정했다. pdf에 나와있는 BNF Grammar을 보고 문법을 수정했다.

```
5
6 %token IF ELSE INT RETURN VOID WHILE THEN END REPEAT UNTIL READ WRITE
7 %token ID NUM
8 %token LBRACE RBRACE LCURLY RCURLY LT LE GT GE NE COMMA
9 %token ERROR
0 %token ASSIGN EQ PLUS MINUS TIMES OVER LPAREN RPAREN SEMI
1
```

: 맨 위쪽에 %token을 통해 token을 추가했다.

이 때

```
identifier
    : ID
      {
        $$ = newExpNode(IdK);
        $$ -> attr.name = copyString(tokenString);
        //savedName = copyString(tokenString);
        //savedLineNo = lineno;
      }
    ;


number
    : NUM
      {
        $$ = newExpNode(ConstK);
        $$->attr.val=atoi(tokenString);
        ///savedLineNo=lineno;
        //savedNumber = atoi(tokenString);
        //savedLineNo=lineno;
      }
    ;
```

: ID와 NUM은 따로 값을 저장해주는 문법을 지정해주었다. 각각 노드를 새로 생성한다. 모든 identifier가 나오는 문법은 IdK 노드를 생성해 Id: 문자를 출력한다. (number도 ConstK노드를 생성한다.)

그 외의 과제 길이 상 declaration부분 코드만 첨부한다.

```
program
    : declaration_list
      { savedTree = $1; }
    ;


declaration_list
    : declaration_list declaration
      {
        YYSTYPE t = $1;
        if ( t != NULL )
        {
            while ( t -> sibling != NULL )
                t = t -> sibling;
            t -> sibling = $2;
            $$ = $1;
        }
        else
            $$ = $2;
      }
    | declaration
      { $$ = $1; }
    ;


declaration
    : var_declaration
      { $$ = $1; }
    | fun_declaration
      { $$ = $1; }
    ;
```

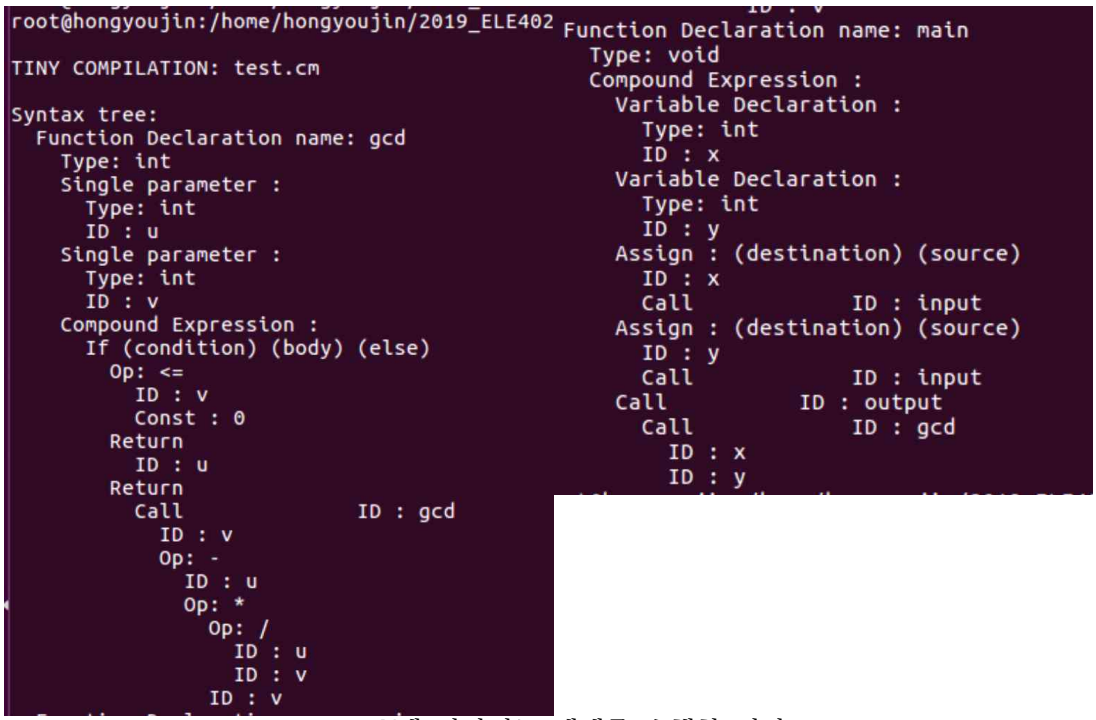
(5) util.c

각 enum별로 해당되는 내용을 print한다. 과제 레포트 길이상 stmt부분만 첨부한다.

```
switch (tree -> kind.stmt)
{
    case AssignK:
        fprintf(listing, "Assign : (destination) (source)\n");
        break;
    case CompK:
        fprintf(listing, "Compound Expression :\n");
        break;
    case SelK:
        if(tree->child[2]==NULL){
                fprintf(listing, "If (condition) (body)\n");
        }else{
                fprintf(listing, "If (condition) (body) (else)\n");
        }
        break;
    case IterK:
        fprintf(listing, "While\n");
        break;
    case RetK:
        fprintf(listing, "Return\n");
        break;
    default:
        fprintf(listing, "Unknown ExpNode kind\n");
        break;
}
```

: SelK는ㄴ if를 의미하는데 이때 child[2]가 null인지를 확인해 else가 있는지 없는지 확인한다.

3. 과제 결과

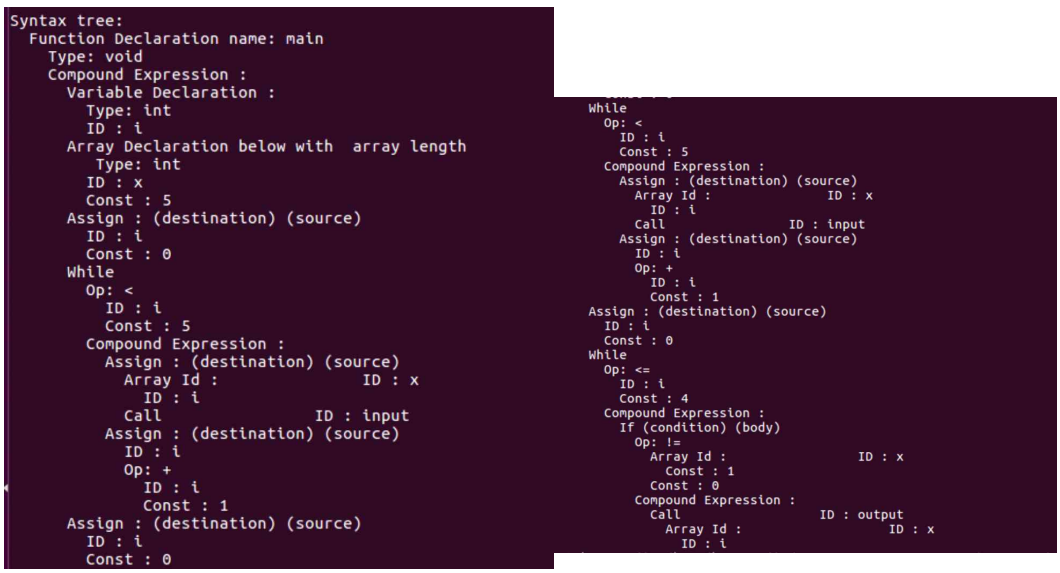


```
root@hongyoujin:/home/hongyoujin/2019_ELE402

TINY COMPILATION: test.cm

Syntax tree:
  Function Declaration name: gcd
    Type: int
    Single parameter :
      Type: int
      ID : u
    Single parameter :
      Type: int
      ID : v
    Compound Expression :
      If (condition) (body) (else)
        Op: <=
          ID : v
          Const : 0
        Return
          ID : u
        Return
          Call              ID : gcd
            ID : v
            Op: -
              ID : u
              Op: *
                Op: /
                  ID : u
                  ID : v
                ID : v
```

```
Function Declaration name: main
  Type: void
  Compound Expression :
    Variable Declaration :
      Type: int
      ID : x
    Variable Declaration :
      Type: int
      ID : y
    Assign : (destination) (source)
      ID : x
      Call            ID : input
    Assign : (destination) (source)
      ID : y
      Call            ID : input
    Call          ID : output
      Call            ID : gcd
        ID : x
        ID : y
```

<pdf에 나와있는 예제를 수행한 결과>

: ID를 계속 노드생성을 했기 때문에 pdf와 출력이 다를 수 있습니다.(예를들어 Single parameter같은 경우 name으로 출력하지않고 ID로 따로 출력)



```
Syntax tree:
  Function Declaration name: main
    Type: void
    Compound Expression :
      Variable Declaration :
        Type: int
        ID : i
      Array Declaration below with  array length
         Type: int
        ID : x
        Const : 5
      Assign : (destination) (source)
        ID : i
        Const : 0
      While
        Op: <
          ID : i
          Const : 5
        Compound Expression :
          Assign : (destination) (source)
            Array Id :              ID : x
              ID : i
            Call              ID : input
          Assign : (destination) (source)
            ID : i
            Op: +
              ID : i
              Const : 1
      Assign : (destination) (source)
        ID : i
        Const : 0
```

```
While
  Op: <
    ID : i
    Const : 5
  Compound Expression :
    Assign : (destination) (source)
      Array Id :              ID : x
        ID : i
      Call            ID : input
    Assign : (destination) (source)
      ID : i
      Op: +
        ID : i
        Const : 1
Assign : (destination) (source)
  ID : i
  Const : 0
While
  Op: <=
    ID : i
    Const : 4
  Compound Expression :
    If (condition) (body)
      Op: !=
        Array Id :              ID : x
          Const : 1
        Const : 0
      Compound Expression :
        Call            ID : output
          Array Id :              ID : x
            ID : i
```

: 위 사진은 project1에서 블랙보드에 올려 주신 test를 출력한 내용입니다. array declaration과 var을 어떻게 출력하는지 확인할 수 있습니다.